METHOD, SYSTEM AND PROGRAM PRODUCTS FOR EFFICIENTLY LOCKING RESOURCES OF A GLOBAL DATA REPOSITORY

Cross-Reference to Related Applications

This application contains subject matter which is related to the subject matter of the following applications, each of which is assigned to the same assignee as this application and filed on the same day as this application. Each of the below listed applications is hereby incorporated herein by reference in its entirety:

Sub A1

"Method, System And Program Products For Modifying Globally Stored Tables Of A Client-Server Environment," Uceda-Sosa et al., (Docket No. POU9-2000-0016-US1), Serial No. _____, filed _____;

Sub B1

"Method, System And Program Products For Concurrent Write Access To A Global Data Repository," Uceda-Sosa et al., (Docket No. POU9-2000-0012-US1, Serial No. _____, filed _____; and

"Method, System And Program Products For Concurrently Accessing A Global Data Repository By Multithreaded Clients," Uceda-Sosa et al., (Docket No. POU9-2000-0019-US1, Serial No. _____, filed _____.

Technical Field

This invention relates, in general, to a distributed computing environment, and in particular, to managing the

locking of resources located in a global repository of the distributed computing environment.

Background Art

Distributed systems are highly-available, scalable systems that are utilized in various situations, including those situations that require a high-throughput of work or continuous or nearly continuous availability of the system.

Typically, users of a distributed system desire access to and modification of data residing in a global repository accessible by one or more users of the system. In order to prevent corruption of the data, techniques are needed to manage the access and modification of the data.

For instance, assume that a repository has a hierarchical topology having various levels. Previously, in order to lock requested resources of the repository, hierarchical locks (e.g., read/write locks) were used. Hierarchical locks require, however, the employment of a lock at each level of the hierarchy. Thus, if the topology has five (5) levels, and a resource on the fifth level is to be locked, then five locks are needed.

The employment and management of so many locks degrades system performance. Thus, a need still exists for an efficient locking facility that enables concurrent access to resources of a data repository without requiring locks on each level of the data repository topology.

Summary of the Invention

The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a method of managing the locking of resources of a data repository. The method includes, for instance, determining a relationship between a plurality of resources of the data repository, wherein the relationship is at least one of a containment-based relationship and a reference-based relationship; and locking at least one resource of the plurality of resources based on the relationship.

System and computer program products corresponding to the above-summarized methods are also described and claimed herein.

Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

FIG. 1 depicts one example of a distributed computing environment incorporating and using one or more aspects of the present invention;

FIG. 2 depicts one example of a plurality of nodes of FIG. 1 coupled to a global data repository, in accordance with an aspect of the present invention;

FIG. 3 depicts one example of a hierarchical topology of the global data repository of FIG. 2, in accordance with an aspect of the present invention;

FIG. 4 depicts another example of the hierarchical topology of FIG. 3 annotated with relationships, in accordance with an aspect of the present invention; and

FIG. 5 depicts one embodiment of the logic associated with managing the locking of resources of a global data repository, in accordance with an aspect of the present invention.

Best Mode for Carrying Out the Invention

In accordance with one or more aspects of the present invention, a locking facility is provided that enables concurrent access to a complex data structure, while minimizing the lock acquisition necessary to access a particular resource of the complex data structure.

The complex data structure (e.g., a global data repository) is to be concurrently accessible by a plurality of users executing within a distributed computing environment. One example of a distributed computing environment incorporating and using aspects of the present invention is depicted in FIG. 1 and described herein. A distributed computing environment 100 includes, for instance, a plurality of frames 102, coupled to one another via a plurality of LAN gates 104. Frames 102 and LAN gates 104 are described in detail herein.

In one example, distributed computing environment 100 includes eight (8) frames, each of which includes a plurality of processing nodes 106. In one instance, each frame includes sixteen (16) processing nodes (each having one or more processors). Each processing node is, for instance, a RISC/6000 computer running AIX, a UNIX based operating system. Each processing node within a frame is coupled to the other processing nodes of the frame, via for example, an internal LAN connection. Additionally each frame is coupled to the other frames via LAN gates 104.

As examples, each LAN gate 104 includes either a RISC/6000 computer, any computer network connection to the LAN, or a network router. However, these are only examples. It would be apparent to those skilled in the relevant art that there are other types of LAN gates, and that other mechanisms can also be used to couple the frames to one another.

The distributed computing environment of FIG. 1 is only one example. It is possible to have more or less than eight frames, or more or less than sixteen nodes per frame. Further, the processing nodes do not have to be RISC/6000 computers running AIX. Some or all of the processing nodes can include different types of computers and/or different operating systems. For example, this invention can be employed with LINUX and/or Windows operating systems. Further, a heterogeneous environment can include and utilize aspects of the invention in which one or more of the nodes and/or operating systems of the environment are distinct from other nodes or operating systems of the environment. The nodes of such a heterogeneous environment interoperate, in that they collaborate and share resources with each other. All of these variations are considered a part of the claimed invention.

At least a plurality of the nodes of the distributed computing environment are capable of sharing resources and data with one another. In order to facilitate the sharing of data, the nodes are coupled to one or more global data repositories 200 (FIG. 2). In one example, each data repository is a relational repository, in which the data (e.g., configuration data) is stored in one or more resources, such as relational tables, that can be concurrently accessed by users of the nodes. Each data repository is remote from the nodes, in this example. However, in other examples, the repository can be local to one or more of the nodes. Further, the repository can be remote or local to users of the repository.

Examples of a data repository and the management of such is described in, for instance, the following U.S. Patent Applications: "Method, System And Program Products For Managing A Clustered Computing Environment," Novaes, et al., Serial No. _____, filed _____; "Method, System And Program Products For Modifying Globally Stored Tables Of A Client-Server Environment," Uceda-Sosa, et al., Serial No. _____, filed _____; "Method, System And Program Products For Concurrent Write Access To A Global Data Repository," Uceda-Sosa, et al., Serial No. _____, filed _____; and "Method, System And Program Products For Concurrently Accessing A Global Data Repository By Multithreaded Clients," Uceda-Sosa, et al., Serial No. _____, filed _____, each of which is hereby incorporated herein by reference in its entirety.

Further details regarding a global data repository are described with reference to FIG. 3. The resources (e.g., tables, directories) of the data repository have hierarchical relationships with one another thus, forming one or more linked data trees 300. Each linked data tree is managed by a repository server coupled to the data repository and can be accessed concurrently by, for instance, logged users. In one example, the main data unit of the repository is a relational table 302, which, in this example, is the smallest lockable data unit of the repository. (In other embodiments however, the smallest lockable data unit can be something other than a table.) Each table includes one or more rows and columns.

Tables exist in the context of at least one directory 304. Directories can include other directories and/or tables. In this example, there is a distinguished directory, called the root directory 306, which is the topmost directory. The root directory cannot be deleted. Since each directory defines a name space, it is desirable at times for the same table or subdirectory to appear under different directories (referred to as parent directories). In this case, each of the parent directories has a hard link to the child resource (e.g., table or subdirectory), for the children will not be removed from the system until they have been deleted from all of the parent directories. This scenario provides the most generic data topology for a hierarchical (directory-oriented) database, because the data tree becomes a graph. Furthermore, this graph changes dynamically as new directories and tables are added or deleted.

Various operations can be performed on the resources of a repository, and these operations include, for instance, create, delete, read and write. When performing these operations, locks are typically employed, such that concurrent access to the data repository is provided. In one example, it is assumed that each resource is equipped with a Read/Write lock.

Since the repository has a hierarchical topology, there are various relationships among the resources of the repository and the locks of the repository. In the example described herein, these relationships are based on object-

oriented concepts. Thus, the examples herein refer to instances and references (which is further described below). For example, an instance of X indicates an instance of Type X, which may include a directory or a table, as examples. Likewise, an instance of Y indicates an instance of Type Y, which also may include a directory or a table, as examples. However, the use of object-oriented concepts is only one example. The facilities described herein are equally applicable to other concepts, such as procedural languages. For example, in the case of procedural languages, instead of employing the terms instances, objects and references, the terms structures and pointers may be used.

The type of locking relationship that exists depends on the particular relationship between the resources. For example, assume there are two resources, an instance of X (a.k.a., X) and an instance of Y (a.k.a., Y) that are independent and global (i.e., accessible from several threads). If an operation desires access to both of these resources, then it sequences their lock acquisition (in the order in which they are mentioned). Thus, the relationship of the resources, as well as the lock acquisition, is considered independently sequenced.

However, in a further example, assume that X is accessed through Y. In this scenario, the acquisition of locks for X ties to the acquisition of locks for Y, also. Thus, the resources and the lock acquisition are dependently sequenced.

There are two cases of dependently sequenced resources discussed herein. The first case relates to containment, and the second case relates to reference, each of which is described in detail below. Again, these relationships (e.g., containment and reference) are described in terms of object-oriented concepts, but this is only one example.

CONTAINMENT:

If there is exactly one reference from X to Y, whenever Y exists, then X is a container for Y, and Y is considered a unireferenced resource. In containment, X is not deleted throughout the life of the process, or in some way, it is guaranteed that instances of Y will not be accessed after X has been destroyed. The locking acquisition for containment is referred to herein as chained locking.

With a chained locking strategy, the particular lock(s) to be obtained depend on, for instance, the operation to be performed, as well as the relationship of the resources, which is containment, in this scenario. Examples of the locks obtained for the various operations that can be performed include the following.

In order to create an instance of Y, a write lock is obtained for X. Further, in order to delete an instance of Y, write locks to both X and Y are obtained. However, in order to read an instance of Y, a read lock of Y is obtained, without the need for obtaining a lock for X.

Similarly, to write to an instance of Y, a write lock to Y is obtained, and no lock for X is needed.

REFERENCE:

If there may be more than one reference from an instance of X to an instance of Y, X is a reference to Y. With a reference relationship, the instance of Y is created on the first reference from an instance of X, and deleted when the last reference has been removed.

For example, if a resource (e.g., a table or a directory) can be included in more than one directory, such a resource is considered multireferenced. As an example Table B of FIG. 4 is multireferenced because it is included in and may be referenced by both Directory A and Directory B. Multireferenced resources are deleted either explicitly (by issuing, for instance, a delete_table request) or implicitly, by deleting its parent directories. When a multireferenced resource is explicitly deleted, it is deleted from all of its directories. A multireferenced resource is implicitly deleted only when it has been deleted from all of its directories. A directory cannot be, in one example, multireferenced from subdirectories (i.e., loops in the reference graph of a directory are forbidden).

The locking acquisition for reference is referred to herein as a reference-based locking strategy. As with the chained locking strategy, the reference-based locking acquisition strategy is also based on the operation to be

performed, as well as the relationship of the resources, which is reference, in this scenario. Examples of the locks obtained for the various operations that can be performed include the following:

In order to delete X, a write lock to X is obtained. Further, in order to create Y, a write lock to X is obtained (this also imports a first reference from the instance of X to Y). For a deletion of Y, a write lock to all of the instances of X that hold a reference to Y is obtained. This implies that all the references are removed. For a read of Y, a read lock to an instance of X and a read lock to Y are obtained. For a write of Y, a read lock to an instance of X and a write lock to Y are obtained. In order to set/remove X's reference to Y, a write lock to X is obtained. If the reference is bidirectional, then a write lock to Y is also obtained.

In accordance with an aspect of the present invention, relationship information is stored, for instance, within each directory (in the client and/or server). In particular, each parent directory includes information that describes the relationship of the directory to its resources. For instance, each parent directory includes for each of its resources, a reference to the resource; and the characterization of the relationship with the resource as either containment in the case of unireferenced resources and reference in the case of multireferenced resources (see FIG. 4). Further, each resource keeps a reference count of the number of references made to it. By default, this count

is 1. (In other embodiments, other information may be stored in the resources. Further, the relationship information may be stored in less than all of the parent directories or it may be stored in one or more resources.)

As described above, in accordance with an aspect of the present invention, locking of a resource is based upon, for instance, the relationship of the resource with one or more other resources of the repository. Thus, prior to selecting a locking strategy, STEP 502 (FIG. 5), the relationship of the resource is defined, STEP 500.

In one embodiment, in order to establish whether a container or reference relationship is appropriate, and thus, the appropriate locking strategy, the following policies are followed:

1) The root directory is considered a container of its tables and subdirectories, since it cannot be deleted and referenced.

2) Other directories are considered containers of their tables, as long as they are not multireferenced.

3) When acquiring locks for a set of resources, the same order is followed: First, acquire write locks by lexicographical order, then read locks by lexicographical order.

4) Assume a hard link is made from a directory, DirA, to a resource, Resource X, (in the case of a directory, assume that it is a legal link) residing in DirB. Suppose that X was unireferenced prior to the link from DirA. The following locking strategy is used:

(*) Lock both DirA and DirB for write (according to the order specified in 3 above). Update the links of DirA and DirB to X to reference. Update the reference count for X.

5) When a hard link is made from a directory, DirA, to a resource, Resource X, which is multireferenced, the reference strategy described above is followed. Further, the reference count for X is updated.

6) Assume that a hard link is removed from a directory, DirA, to Resource X, which is multireferenced. The strategy for deletion of the link in the reference locking strategy is followed. Further, the reference count for X is updated. However, if the count is reduced from 2 to 1, an exception is made. In particular, suppose the another competing link is from DirB. Then, both DirA and DirB are locked (by releasing all previous locks and starting from scratch). Further, the link characterization of DirB is modified from reference to container. Thereafter,

the link is removed from DirA, and the reference count of X is updated.

7) When accessing a resource, the locking policies of all the links leading to it are respected.

Described above are various scenarios for acquiring locks based on relationships, such that maximum concurrency access to the global data repository is facilitated, while minimizing the number of locks obtained. Further scenarios are described below, in which the improved speed in obtaining lock acquisitions for read/write operations (especially for unireferenced resources) is illustrated.

Examples of obtaining locks based on the relationship of the resources is described below:

(1) For a read/write operation on DirC (see FIG. 4): Lock DirC for read or write, instead of acquiring a set of locks corresponding to all of the resources in the path; thus, significantly reducing the number of locks acquired.

(2) For a write operation on TabB (see FIG. 4): A read lock to DirB and a write lock to TabB are obtained.

As described above, advantageously, maximum concurrency access to a data repository is provided, while minimizing

the lock acquisition necessary to access a resource (e.g., table or directory) of the repository.

In one embodiment, the repository server, which is coupled to and manages the data repository, is multithreaded. Thus, one or more aspects of the present invention advantageously provide the ability to read or write a set of related resources in a dynamic graph that can be globally accessed in a multithreaded environment.

Although the description above focused on one data repository, the capabilities of the present invention extend to a plurality of data repositories of a computing environment. One or more of the repositories can be local and/or remote to the users accessing the repositories.

Further, one or more aspects of the present invention are applicable to homogeneous systems, as well as heterogeneous systems. As one example, capabilities are provided to facilitate the interoperability of the systems of a heterogeneous environment.

The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

The flow diagrams depicted herein are just examples. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.